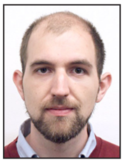
Some Routes Are More Default than Others

JONATHON ANDERSON



Jonathon Anderson has been an HPC Sysadmin since 2006 and believes that everything would be a lot easier if we just spent more time figuring out the correct way to do things. He's currently serving as HPC Engineer at the University of Colorado, and hopes to stick around Boulder for a long time to come.
jonathon.anderson@colorado.edu

Typical IP-networked hosts are configured with a single default route. For single-homed hosts the default route defines the first destination for packets addressed outside of the local subnet; but for multi-homed hosts the default route also implicitly defines a default interface to be used for all outbound traffic. Specific subnets may be accessed using non-default interfaces by defining static routes; but the single default route remains a "single point of failure" for general access to other and Internet subnets. The Linux kernel, together with the iproute2 suite [1], supports the definition of multiple default routes distinguished by a preference metric. This allows alternate networks to serve as failover for the preferred default route in cases where the link has failed or is otherwise unavailable.

Background

The CU-Boulder Research Computing (RC) environment spans three datacenters, each with its own set of special-purpose networks. Public-facing hosts may be accessed through a 1:1 NAT or via a dedicated "DMZ" VLAN that spans all three environments. We have historically configured whichever interface was used for inbound connection from the Internet as the default route in order to support responses to connections from Internet clients; but our recent and ongoing deployment of policy routing (as described in the summer 2016 issue of *;login:*) removes this requirement.

All RC networks are capable of routing traffic with each other, the campus intranet, and the greater Internet, so we more recently prefer the host's "management" interface as its default route as a matter of convention; but this unnecessarily limits network connectivity in cases where the default interface is down, whether by link failure or during a reconfiguration or maintenance process.

The Problem with a Single Default Route

The simplest Linux host routing table is a system with a single network interface.

```
# ip route list
default via 10.225.160.1 dev ens192
10.225.160.0/24 dev ens192  proto kernel  scope link  src 10.225.160.38
```

Traffic to hosts on 10.225.160.0/24 is delivered directly, while traffic to any other network is forwarded to 10.225.160.1. In this case, the default route eventually provides access to the public Internet.

```
# ping -c1 example.com
PING example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34: icmp_seq=1 ttl=54 time=24.0 ms

--- example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 24.075/24.075/24.075/0.000 ms
```

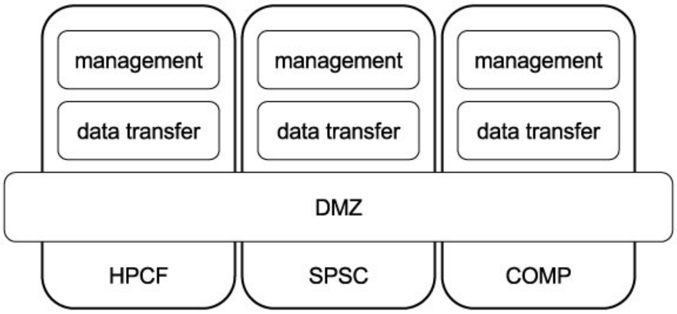


Figure 1: The CU-Boulder Research Computing Science Network, with subnets in three datacenters

A dual-homed host adds a second network interface and a second link-local route, but the original default route remains.

```
# ifup ens224 && ip route list
default via 10.225.160.1 dev ens192
10.225.160.0/24 dev ens192  proto kernel  scope link  src
10.225.160.38
10.225.176.0/24 dev ens224  proto kernel  scope link  src
10.225.176.38
```

The new link-local route provides access to hosts on 10.225.176.0/24, but traffic to other networks still requires access to the default interface as defined by the single default route. If the default route interface is unavailable, external networks become inaccessible, even though identical routing is available via 10.225.176.1.

```
# ifdown ens192 && ping -c1 example.com; ifup ens192
connect: Network is unreachable
```

Attempts to add a second default route fail with an error message (in typically unhelpful iproute2 fashion), implying that it is impossible to configure a host with multiple default routes simultaneously.

```
# ip route add default via 10.225.176.1 dev ens224
RTNETLINK answers: File exists
```

It would be better if the host could select dynamically from any of the physically available routes, but without an entry in the host's routing table directing packets out the ens224 "data" interface, the host will simply refuse to deliver the packets.

Multiple Default Routes and Routing Metrics

The RTNETLINK error above indicates that the ens224 "data" route cannot be added to the table because a conflicting route already exists—in this case, the ens192 "management" route. Both routes target the "default" network, which would lead to non-deterministic routing with no way to select one route in favor of the other.

However, the Linux routing table supports more attributes than the "via" address and "dev" specified in the above example. Of use here, the "metric" attribute allows us to specify a preference number for each route.

```
# ip route change default via 10.225.160.1 dev ens192 metric 100
# ip route add default via 10.225.176.1 dev ens224 metric 200
# ip route flush cache
```

The host will continue to prefer the ens192 "management" interface for its default route due to its lower metric number, but if that interface is taken down, outbound packets will automatically be routed via the ens224 "data" interface.

```
# ifdown ens192 && ping -c1 example.com; ifup ens192
PING example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from example.com (93.184.216.34): icmp_seq=1 ttl=54
time=29.0 ms

--- example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 29.032/29.032/29.032/0.000 ms
```

Persisting the Configuration

This custom-routing configuration can be persisted in the Red Hat "ifcfg" network configuration system by specifying a METRIC number in the ifcfg- files. This metric will be applied to any route populated by DHCP or by a GATEWAY value in the ifcfg- file or /etc/sysconfig/network file.

```
# grep METRIC= /etc/sysconfig/network-scripts/ifcfg-ens192
METRIC=100

# grep METRIC= /etc/sysconfig/network-scripts/ifcfg-ens224
METRIC=200
```

Alternatively, routes may be specified using route- files. These routes must define metrics explicitly.

```
# cat /etc/sysconfig/network-scripts/route-ens192
default via 10.225.160.1 dev ens192 metric 100

# cat /etc/sysconfig/network-scripts/route-ens224
default via 10.225.176.1 dev ens224 metric 200
```

Alternatives and Further Improvements

The NetworkManager service in RHEL 7.x handles multiple default routes correctly by supplying distinct metrics automatically; but, of course, specifying route metrics manually allows you to control which route is preferred explicitly.

I continue to wonder whether it might be better to go completely dynamic and actually run OSPF [2] on all multi-homed hosts. This should—in theory—allow our network to be even more automatically dynamic in response to link availability, but this may be too complex to justify in our environment.

There's also potential to use all available routes simultaneously with weighted load-balancing, either per-flow or per-packet [3]. This is generally inappropriate in our environment but could be preferable in an environment where the available networks are definitively general-purpose.

```
# ip route equalize add default \
    nexthop via 10.225.160.1 dev ens192 weight 1 \
    nexthop via 10.225.176.1 dev ens224 weight 10
```

Conclusion

We've integrated a multiple-default-route configuration into our standard production network configuration, which is being deployed in parallel with our migration to policy routing. Now the default route is specified not by the static binary existence of a single `default` entry in the routing table but by an order of preference for each of the available interfaces. This allows our hosts to remain functional in more failure scenarios than before, when link failure or network maintenance makes the preferred route unavailable.

References

[1] iproute2: http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2.

[2] OSPFv2: https://www.ietf.org/rfc/rfc2328.txt.

[3] Policy Routing: http://www.policyrouting.org/PolicyRoutingBook/ONLINE/CH05.web.html.